ER2FGSM: Efficient and Robust Adversarial Attacks Against Deep Reinforcement Learning

Abstract

7 pages + 2 pg ref.; full paper deadline: 9/8

The high performance of Deep Reinforcement Learning (DRL) algorithms in complex tasks has increased the need to ensure the robustness of the DRL against advanced adversarial attacks. However, compared to a large volume of studies exploring adversarial attacks to disrupt Deep Neural Networks (DNN), adversarial attacks to disrupt DRL have been significantly less explored. The existing adversarial algorithms for DRL have focused on attacking a DRL agent mainly disrupting its decision making. However, little work has explored in injecting perturbation (adversarial example) attacks in input data in the context of DRL settings. In this paper, we propose efficient and robust perturbation-based adversarial attacks to disrupt the states in the DRL process, called *Efficient and Robust Randomized Fast Gradient Sign Method* (ER2FGSM). We enhanced the Randomized Fast Gradient Sign Method (RFGSM) that was previously studied under DNN settings so that it can be applied under DRL settings. We considered targeted or non-targeted attacks under DRL with or without defenses to investigate the efficiency and robustness of the ER2FGSM. We conducted extensive experiments using Deep Q-Network (DQN) agent (i.e., DRL agent) playing an Atari Pong game where our ER2FGSM is compared against the five different state-of-the-art DRL attacks in terms of attack success rate, attack execution time, and robustness under defenses. Our results proved that our proposed ER2FGSM attack is 17 times faster than the state-of-the-art Carlini & Wagner method while showing highly comparable attack success rate and robustness.

Introduction

Deep Reinforcement Learning (DRL) algorithms learn policies to guide DRL agents to take optimal actions based on environment state. These algorithms achieved really high performance on the various complex as well as critical tasks, such as robotics (Amarjyoti 2017), autonomous vehicles (Kiran et al. 2021; Ferdowsi et al. 2018), resource allocation (Yoon et al. 2021b), intrusion response systems (Yoon et al. 2021a), various cybersecurity problems (Basori and Malebary 2020), or networking and communication problems (Luong et al. 2019). As DRL has been used to solve various problems as above, adversaries aiming to disrupt DRL process and misleading the DRL agent's decision-making have been recognized as a serious issue.

A policy, a probabilistic distribution of actions by the DRL agent, is often learned by Deep Neural Networks (DNN) for the approximation of the action-value function. The vulnerabilities of the DNN to adversarial attacks have been significantly studied (Szegedy et al. 2013; Goodfellow, Shlens, and Szegedy 2014a; Yuan et al. 2019) to mitigate the impact when they are exploited by adversaries. Common adversarial examples include adversarial perturbations imperceptible to human but fooling DNNs easily in the testing or deploying stage (Yuan et al. 2019). Various attacks and defenses have been proposed for supervised DNN applications, such as image classification (Goodfellow, Shlens, and Szegedy 2014a) and natural language processing (Alzantot, Balaji, and Srivastava 2018). However, adversarial attacks and defenses are largely unexplored in DRL-based settings. DRL has numerous critical applications in safety and security and accordingly highlights the importance of robust DRL. For robust DRL, there is a prerequisite of developing efficient, robust adversarial attacks which can be used to evaluate the robustness of defense mechanisms.

Adversarial attacks in DRL based setting are developed generally by answering two major questions: (1) *How to attack?*; and (2) *When to attack?*. The first *how-to-attack* question is related to what perturbation method to use for attacking the state during an episode. The second *when-to-attack* question is related to identify an optimal time to attack during an episode. In this work, we focus on answering *how-to-attack* by proposing the so called *Efficient and Robust Randomized Fast Gradient Sign Method*, namely ER2FGSM. To be specific, **the goal of this work** is to develop robust and fast attacks by generating effective adversarial states in DRL settings. In addition, we validate the performance of the proposed ER2FGSM by comparing it against those of the state-of-the-art adversarial attacks under DRL with or without defenses where the attacker can perform targeted or non-targeted attacks, in terms of attack success rate (ASR), average attack execution time per perturbation (AET), and average reward (AR) by the attacker and the defender (i.e., DRL agent).

We made the following **key contributions** in this work:

1. We developed ER2FGSM which can generate fast and robust adversarial perturbations to compromise a DRL

agent where the perturbation attack is targeted or non-targeted and Deep Q-Network (DQN) is used as a DRL algorithm playing an Atari Pong game. ER2FGSM is an enhanced version of RFGSM (Tramèr et al. 2017) and Iterative FGSM (I-FGSM) (Kurakin et al. 2016) which can efficiently attack the states of the DRL process with robustness. However, in our ER2FGSM, instead of identifying an optimal time to attack in the DRL process, we considered attacking on all steps during an episode. This approach allowed us to evaluate the effectiveness (i.e., ASR) of our proposed perturbation attack on all situations during the episode. In addition, this enabled the fair comparison with other existing perturbation attacks.

2. We considered State Adversarial DQN (SA-DQN) (Zhang et al. 2020) and Robust ADversarIAL Loss DQN (RADIAL-DQN) (Oikarinen, Weng, and Daniel 2020) as more robust defenses in the DQN and investigated the effect of the defenses in DRL on the robustness of the proposed ER2FGSM attack in ASR, compared to the five baseline attacks.
3. To validate the outperformance of the proposed ER2FGSM, we conducted extensive performance analysis by comparing it with the five state-of-the-art adversarial examples in DRL, including (Carlini and Wagner 2017; Goodfellow, Shlens, and Szegedy 2014b; Madry et al. 2017; Dong et al. 2018; Xie et al. 2019) in terms of ASR, AET, and AR. Among the five existing adversarial examples, we devised two baseline examples, which are more robust and scalable, by extending the Interative FGSM (DI-FGSM) (Xie et al. 2019) and Momentum Iterative FGSM (MI-FGSM) (Dong et al. 2018) used in DNNs into a DRL setting. Our results showed that our proposed ER2FGSM significantly outperformed the existing counterparts in AET while comparably maintaining ASR and AR.

The rest of the paper is structured as follows. We will discuss the related work in terms of attacks in DNNs and DRL. In the Preliminaries, we will provide the brief overview of DRL with DQN and a set of adversarial states generation methods, which are considered in our ER2FGSM or compared against ER2RFGSM. Then, we formulate our problem statement along with the expected cumulative reward functions by the defender and attacker. In the Proposed Approach, we describe our proposed ER2FGSM. In the Experiment Setup, we describe metrics, comparing schemes, key design parameters and their default values used, and the environmental setup. In the Experiments and Results, we demonstrated the results from the extensive comparative study and discuss their general trends. In the Conclusion and Future Work, we summarized the key findings and the future work directions. can drop this if no room

Related Work

In this section, we provide the brief overview of the state-of-the-art adversarial attacks in Deep Neural Networks (DNN), which is mainly used for supervised deep learning. In addition, we briefly discuss existing adversarial attacks to disrupt DRL process and identify the differences between our proposed ER2FGSM and their approaches.

Backdoor attacks in Deep Neural Networks (DNNs). Szegedy et al. (2013) initially presented an attack with small perturbations that can lead to misclassifications. Later, various attacks and defenses have been proposed and evaluated in terms of scalability, attack success rate (ASR), and robustness. Goodfellow, Shlens, and Szegedy (2014a) proposed the Fast Gradient Sign Method (FGSM) which was efficient but turned out to be less robust, not guaranteeing 100% success rate. Carlini and Wagner (2017) proposed the Carlini & Wagner (CW) method that guarantees 100% ASR but was slow. Madry et al. (2017) proposed the Projected Gradient Descent (PGD), which [add more explanation on its efficiency and effectiveness on PGD] Naïve FGSM provided the basis to build more sophisticated and better variants of FGSM, such as Randomized Fast Gradient Sign Method (RFGSM) (Tramèr et al. 2017), Diversity Iterative Fast Gradient Sign Method (DI-FGSM) (Xie et al. 2019), and Momentum Iterative Fast Gradient Sign Method (MI-FGSM) (Dong et al. 2018). Currently, MI-FGSM and PGD attacks are considered the most efficient and robust state-of-the-art adversarial examples in DNNs. In particular, RFGSM, DI-FGSM, and MI-FGSM are only designed and evaluated in the context of DNNs and have never been used in DRL settings. In this work, we proposed ER2FGSM by enhancing RFGSM and applied it in DRL settings. In addition, we also refined DI-FGSM and MI-FGSM to be applied in the DRL settings that can be compared against ER2FGSM.

Adversarial Attacks in Deep Reinforcement Learning (DRL). State of the art attacks (Lin et al. 2017; Sun et al. 2020) in DRL has often ignored the question of How to Attack? [I don't understand what you mean; describe (Lin et al. 2017; Sun et al. 2020) with more details; why are they not compared against our method?]. Carlini and Wagner (2017) proposed the Carlini & Wagner (CW) method [briefly explain the CW attacks], which are slow and easily detected. Fast perturbation methods, such as naïve Fast Gradient Sign Method (FGSM) (Goodfellow, Shlens, and Szegedy 2014b) has been used in a DRL setting by (Huang et al. 2017) but the naïve FGSM is easily detectable. The state-of-the-art defenses in DRL settings (Zhang et al. 2020; Oikarinen, Weng, and Daniel 2020; Fischer et al. 2019) have recently challenged the robustness of PGD-based attacks (Madry et al. 2017), which were originally considered in DNNs. Therefore, there is a critical need of developing scalable, effective, and robust state perturbation attacks in DRL settings, which is the goal of our paper.

Preliminaries

In this section, we provide the overview of the DRL and adversarial states generation methods considered in this work.

Deep Reinforcement Learning (DRL)

Reinforcement learning (RL) algorithms optimize the expected cumulative reward by training a policy π. This policy can be deterministic or probabilistic function that maps each state s to an action a, $\pi : S \rightarrow A$, where S and A

are state and action spaces, respectively. In DRL, this plicy function π is learned by a neural network. Deep Q-networks (DQN) (Mnih et al. 2013) is one of the well-known DRL algorithms. We have evaluated our attacks on the policy trained by DQN.

Deep Q-Networks (DQN) In Q-learning, the Q-value is the expected cumulative discounted reward when action a is taken in state s. DQN is a kind of Q-learning where Q-values are learned by a neural network (NN) while minimizing the squared Bellman error. DQN has the component called *Experience Replay* which helps decrease the high variance due to Q-learning updates. Another component called *Replay Buffer* stores all recent transitions. To mitigate the correlation due to time, random samples are taken from this buffer. DQN takes an action based on the maximum Q-value.

Adversarial States Generation Methods

The performance of the following methods (or variants of them) are compared against that of our proposed ER2FGSM.

Fast Gradient Sign Method (FGSM) FGSM (Goodfellow, Shlens, and Szegedy 2014a) focuses on attack efficiency to design adversarial examples, rather than optimal performance of adversarial examples. For an image x, a perturbed image x' according to FGSM is:

$$x'_{\text{FGSM}} = x + \epsilon \cdot \text{sign}(\nabla \text{loss}(h(x), y_{\text{true}})) \tag{1}$$

where t is a target label and ϵ is a parameter to make the perturbation small enough to be less noticeable. The loss function can be a cross entropy function. FGSM uses the linear approximation of the model and solves the maximization problem in a closed form, which makes this method very fast. There are three kinds of norm constraints, L_0, L_1, and L_∞, which are used in the literature to constrain the perturbation to be undetectable. For n to be zero, one or infinite norm, the constraint can be presented by:

$$||x - x'||_n < \epsilon. \tag{2}$$

Carlini & Wagner Method (CW) Carlini and Wagner (2017) presented three attacks granteeing 100% ASR; however, they are much slower than naïve FGSM (Goodfellow, Shlens, and Szegedy 2014b). CW optimizes an objective function in a multi step process to generate minimal and optimal perturbations, which makes the perturbation generation very slow.

Projected Gradient Descent (PGD) Madry et al. (2017) proposed PGD, which is a multi-step variant of FGSM, utilizing the projected gradient descent on the negative loss function. This is also the attempt to solve the inner maximization problem. Starting with $x'_0 = 0$, on every iteration, it performs:

$$x'_i = \prod_{x+\delta} (x'_{i-1} + (\alpha \cdot \text{sign}(\nabla \text{loss}_{F,t}(x'_{i-1})))). \tag{3}$$

Momentum Iterative FGSM (MI-FGSM) Dong et al. (2018) proposed MI-FFSM, which is a variant of FGSM that integrates the momentum on each step of an iterative FGSM to escape from poor local maxima and stabilize update directions. Starting with $x'_0 = 0$, on every iteration, it performs:

$$g_i = \mu \cdot g_{i-1} + \frac{\nabla \text{loss}_{F,t}(x'_{i-1})}{||\nabla \text{loss}_{F,t}(x'_{i-1})||_1}, \tag{4}$$

$$x'_i = x'_{i-1} - \text{clip}_\epsilon(\alpha \cdot \text{sign}(g_i)), \tag{5}$$

where μ is the decay factor and g_i is the accumulated gradient at iteration i.

Diversity Iterative FGSM (DI-FGSM) In the I-FGSM method, there is an overfitting problem if the number of steps/iterations increases. Xie et al. (2019) proposed DIFGSM to solve the overfitting problem in I-FGSM. At each iteration, unlike the traditional methods which maximize the loss function directly w.r.t. the original inputs, DIFGSM applies random and differentiable transformations (e.g., random resizing, random padding) to the input images with probability p and maximize the loss function w.r.t. these transformed inputs. This method is also similar to RFGSM as they both try to generate hard and diverse input patterns. This was primarily developed for blackbox settings to preserve generalizability and diversity. This is also similar to I-FGSM with the addition of transformation T. Starting with $x'_0 = 0$, on every iteration, it performs:

$$x'_i = x'_{i-1} - \text{clip}_\epsilon(\alpha \cdot \text{sign}(\nabla \text{loss}_{F,t}(T(x'_{i-1})))). \tag{6}$$

We leveraged the following two methods (i.e., I-FGSM and RFGSM) to develop our ER2FGSM.

Iterative Fast Gradient Sign Method (I-FGSM) Kurakin et al. (2016) proposed I-FGSM, a variant of FGSM, taking multiple small steps of size α in the direction of the sign of gradient. However, FGSM takes only one step of size ϵ to make perturbation small enough. This method also clips the result of multiple steps by ϵ. I-FGSM outperformed FGSM. Starting with $x'_0 = 0$, on every iteration, it performs:

$$x'_i = x'_{i-1} - \text{clip}_\epsilon(\alpha \cdot \text{sign}(\nabla \text{loss}_{F,t}(x'_{i-1}))), \tag{7}$$

where α is a random step.

Randomized FGSM (RFGSM) A single step FGSM method has a problem of converging to degenerate a global minimum. I-FGSM obfuscates a linear approximation of the loss due to small steps. Due to these problems, they generate weak perturbations, which can be easily defended. To tackle this problem, Tramèr et al. (2017) presented RFGSM, which adds a small random step to such single step attacks, in order to escape the non-smooth vicinity of the data point before linearizing the model's loss. Single Step RFGSM is computationally efficient and has a high ASR than I-FGSM in DNNs. It performs:

$$x^{\text{adv}}_{R+FGSM} = x' + (\epsilon - \alpha) \cdot \text{sign}(\nabla_{x'} \text{loss}(x', y_{\text{true}})), \tag{8}$$

where

$$x' = x + \alpha \cdot \text{sign}(\mathcal{N}(0^d, I^d)). \tag{9}$$

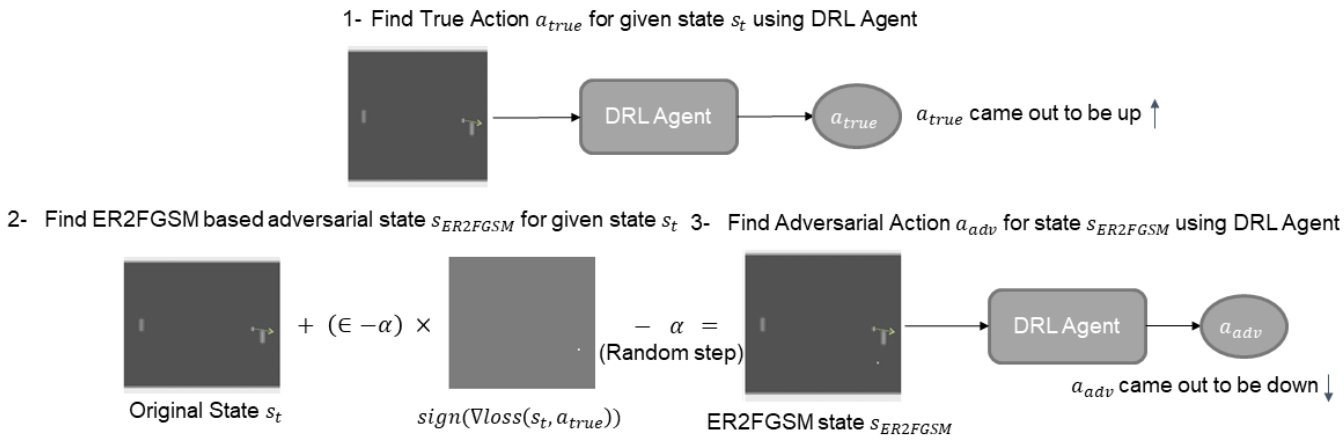

Figure 1: The overview of ER2FGSM: ER2FGSM attacks on a single frame during an episode to compromise a DRL agent following (a) ...; (b) ...; and (c) .... For the corresponding figure to each number, please make them (a), (b), and (c). Then, explain them them (i.e., (a), (b), and (c)) in this caption. add each parameter's meaning in this caption for better readability; describe each notation's meaning

Problem Statement

A DRL agent interacts with an environment and learns policy π to choose action a given a state S. This policy π can be a probabilistic model $\pi(\mathrm{s}, \mathrm{a}) \sim [0, 1]$, which gives the probability of taking action a given a state s. The π can also be a deterministic model where we can obtain action a directly from policy function π given the state $s : a = \pi(\mathrm{s})$. The goal of the DRL agent is to maximize the cumulative reward R_o by learning a policy π^*. The reward that DRL agent aims to maximize is the expected discounted reward until the next $T - 1$ time steps, which is represented by:

$$R_o = \sum_{t=0}^{T-1} E_{a_t \sim u(s_t)} \Big[\gamma^t r(s_t, a_t) \Big]. \tag{10}$$

Instead of maximizing this reward, an adversary aims to reduce reward R_o by adding perturbation δ_t into the agent's observation s_t to mislead the agent to take adverse action a_t. The adversary has to generate perturbation δ as small as possible to be undetected. Out of all time steps, the adversary may or may not choose to add perturbation δ to the state s_t based on his strategy. In this way, the adversary's expected cumulative reward is given by:

$$R_{\text{adv}} = \sum_{t=0}^{T-1} E_{a_t \sim u(s_t + x_t \delta_t)} \Big[\gamma^t r(s_t, a_t) \Big], \tag{11}$$

where x_t at given time t is 0 if the adversary does not inject any perturbation; otherwise, it returns 1.

We need to find the perturbations $\delta_0, \delta_1, \ldots, \delta_{T-1}$ for all time steps during an episode in order to compromise the DRL agent to take adversarial actions $a_0^{\text{adv}}, a_2^{\text{adv}}, \ldots, a_{T-1}^{\text{adv}}$. These adversarial actions should then minimize the reward of the DRL agent, R_o, and the reward of the DRL agent under defense, R_{defense}. We also want to find each perturbation δ_t in the minimum time possible.

Proposed Approach: ER2FGSM

We propose Efficient and Robust Randomized Fast Gradient Sign Method (ER2FGSM) to find robust and effective perturbations in minimum time possible. ER2FGSM is designed by combining RFGSM and I-FGSM and extending the combination from a DNN setting to a DRL setting.

For the extension to the DRL setting, we consider true label y_{true} as a vector of weights for each action produced by the policy of DRL agent, which is presented by:

$$a_{\text{true}} = \text{DRLAgent}(s_{\text{true}}), \tag{12}$$

where s_{true} is the original non-adversarial state and a_{true} is the original non-adversarial action given by the DRL agent.

In ER2FGSM, at each step i, we are calculating the adversarial state $s_i^{ER2FGSM}$ by using the random step α added to the previous adversarial state $s_{i-1}^{ER2FGSM}$ and then clipping the gradient ∇ by step size $\epsilon - \alpha$. Starting with $s_0 = s_{\text{true}}$, on every iteration i, it performs:

$$s_i^{ER2FGSM} = s' + (\epsilon - \alpha) \cdot \text{sign}(\nabla_{s'} \text{loss}(s', a_{\text{true}})), \tag{13}$$

where

$$s' = s_{i-1}^{ER2FGSM} + \alpha \cdot \text{sign}(\mathcal{N}(0^d, I^d)). \tag{14}$$

Equation (13) means that we are moving policy π away from an optimal action by using the direction of the gradient. The significance of making it iterative is to take multiple random steps and multiple gradient direction steps. Multiple random steps are making this method robust as they allow to escape the non-smooth vicinity of the data point before linearizing the model's loss. Multiple gradient steps are moving the policy away from the optimal action, contributing to generating good ASR. Attack effectiveness on even fewer gradient steps and computationally inexpensive calculation of gradients make ER2FGSM fast enough. ER2FGSM is illustrated in Figure 1.

As discussed earlier, our proposed ER2FGSM attacks on all steps during an episode, which allows the evaluation of

the effectiveness and efficiency of ER2FGSM on all situations during the episode. In addition, this enables comparison of our ER2FGSM and the other state-of-the-art attack methods. We always consider $x_t = 1$ as we attacked on all number of steps during the episode by $\sum_{t=0}^{T-1} x_t = \mathrm{N}$ where N is the same as the length of the episode.

Experimental Setup

Metrics

Following are the metrics used to compare our attacks with existing attacks:

- **Attack Success Rate (ASR)**: This measures the total number attack success over the total number of attack attempts. We consider both targeted attacks and non-targeted attacks. Hence, we measure ASR under non-targeted or targeted attacks as:

$$\mathrm{ASR}_{NT} = \frac{N_{NT}^{AS}}{N_{NT}}, \;\; \mathrm{ASR}_{T} = \frac{N_{T}^{AS}}{N_{T}}, \tag{15}$$

  where N_{NT}^{AS} refers to the total number of success by non-targeted attacks and N_{NT} is the total number of attempts by non-targeted attacks. Similarly, N_{T}^{AS} refers to the total number of success by targeted attacks and N_T is the total number of attempts by targeted attacks. Under non-targeted attacks, a failure indicates the case where the attack is entirely unable to succeed or the DRL agent takes the action maximizing the reward. An attack attempt is defined as an attack on one time step. On the other hand, targeted attacks means generating a perturbation aiming to lead the DRL agent to take a targeted action. For example, in Atari Pong, a targeted attack can aim to make the DRL agent to take a targeted action 'up' at a given point. This, reward reduction is not simply considered as success under targeted attacks if it is not by the perturbation method.
- **Average Attack Execution Time Per Perturbation (AET)**: This captures the average time required to generate one perturbation or a perturbed state. For the respective attacks, targeted or non-targeted, we measure AET by:

$$\mathrm{AET}_{T} = \frac{\mathcal{T}(N_{NTS})}{N_S}, \;\; \mathrm{AET}_{T} = \frac{\mathcal{T}(N_{TS})}{N_S}, \tag{16}$$

  where N_S is the total number of adversarial states, $\mathcal{T}(N_{NTS})$ is the total times elapsed to generate all non-targeted attacks, and $\mathcal{T}(N_{TS})$ is the total times elpased to generate all targeted attacks.
- **Average Reward (AR)**: This measures the average reward as the average of the reward of all episodes. Given N_e be the number of episodes and r_i be the total rewards accumulated during an episode, AR is measured by:

$$\mathrm{AR} = \frac{\sum_{i=0}^{N} r_i}{N_e}. \tag{17}$$

Comparing Schemes

We are using the following perturbation attacks to be compared against our ER2FGSM:

- **Fast Gradient Sign Method (FGSM)** (Goodfellow, Shlens, and Szegedy 2014b): We chose this as our basline because it has been not only extensively used in DNN settings but also has been numerous times used in DRL settings, including in the preliminary DRL-based attack, called *uniform attack* (Huang et al. 2017). Since we are using sophisticated versions of FGSM in ER2FGSM, FGSM serves as a good baseline to observe the performance of ER2FGSM.
- **Carlini & Wagner Method (CW)** (Carlini and Wagner 2017): This method is considered one of the well-known state-of-the-art method in DRL settings guaranteeing 100% ASR. The state-of-the-art end goal-based DRL attacks (Lin et al. 2017; Sun et al. 2020) use CW method to generate targeted perturbations. Hence, it is critical to outperform the CW.
- **Projected Gradient Method (PGD)** (Madry et al. 2017): PGD is not much popular in end goal-based DRL attacks but it is considered to be the robust, fast and successful state-of-the-art attack in DRL settings. Due to its robustness, PGD is usually employed by defenders to test their defenses as in (Zhang et al. 2020; Oikarinen, Weng, and Daniel 2020; Fischer et al. 2019). Therefore, outperforming PGD can prove the robustness of our proposed ER2FGSM.
- **Momentum Iterative Fast Gradient Sign Method (MI-FGSM)** (Dong et al. 2018): We extended MI-FGSM from DNN settings to DRL settings and compared it with our method. MI-FGSM is the state-of-the-art adversarial example in DNN settings. It is considered to be the most robust and effective yet fast technique.
- **Diversity Iterative Fast Gradient Sign Method (DI-FGSM)** (Xie et al. 2019): We also extended DI-FGSM from DNN settings to DRL settings and compared it with our method. We included it as our baseline due to its similarity with ER2FGSM as ER2FGSM also indirectly adds diversity by using the concepts of randomization.

Tuned Parameters

We tuned the following parameters for the optimal setting:

- *Size of a perturbation* (ϵ): It should not be very small or very large, so being optimal to be effective but undetected.
- *Number of steps of perturbation* (m): We need to make the steps as low as possible to reduce the attack execution time while achieving high ASR.
- *Size of the step* (α): It defines the size of a step in a perturbation method taking in iterative methods with $\alpha < \epsilon$.

We have extensively parametrized each attack to identify the optimal parameter values for comparison, which are summarized in Table 1.

Defense Setting in DRL

There are very few defenses proposed for adversarial attacks in the DRL setting. The conventional adversarial training defense has been widely used in DNN settings (Kurakin, Goodfellow, and Bengio 2016; Madry et al. 2017) and DRL settings (Kos and Song 2017; Pattanaik et al. 2017; Behzadan and Munir 2017) as well. However, recently more

Perturbation Method	Number of Steps	ϵ	α
CW	1000	NA	NA
PGD	40	8/255	2/255
DI-FGSM	20	8/255	2/255
MI-FGSM	5	8/255	2/255
FGSM	1	2/255	NA
ER2FGSM	8	16/255	8/255

Table 1: Optimal values of the parameters identified for each perturbation method considered in this work

robust and efficient defense methods were proposed specifically designed for DRL setting, such as Robust ADversarIAl Loss DQN (RADIAL-DQN) (Oikarinen, Weng, and Daniel 2020) and State Adversarial DQN (SA-DQN) (Zhang et al. 2020). We use both RADIAL-DQN and SA-DQN to test the robustness of our attacks against their attack detectability.

State Adversarial DQN (SA-DQN) Zhang et al. (2020) proposed a robust policy regularizer related to the total variation distance or Kullback–Leibler (KL)-divergence on perturbed policies. SA-DQN adds a hinge loss regularizer to Q-learning so that the DQN agents will tend to follow their original actions when there are perturbations in the observation space. It ensures that the agent does not change its preferred action under input perturbations.

Robust ADversarIAl Loss DQN (RADIAL-DQN) RADIAL-DQN (Oikarinen, Weng, and Daniel 2020) improved the robustness of DRL agents by designing the adversarial loss functions with robustness verification bounds during training. It leverages robustness verification bounds to keep the loss as low as possible because low loss leads to better performance of the DRL agent. It primarily parametrizes k to minimize the loss where $L_{adv} = kL_S + (1-k)L_W$. RADIAL-DQN applies robustness verification algorithms on the DQN to obtain the layer-wise output bounds of Q-Networks and uses these output bounds to calculate an upper bound of the original loss function under worst-case adversarial perturbation L_W, given L_S is the standard loss function.

Environmental Setup

We consider a white box attack where an adversary does not have access to the training time. However, the adversary has the test time access of the DRL agent where it knows the network architecture and has access to craft its own adversarial state. For our experiments, we trained Deep Q-Network (DQN) using the implementation of (Mnih et al. 2015) to play an Atari Pong game using Pytorch and Open AI Gym. The loss we have used is the cross entropy loss for gradient-based attacks. Assuming that the attacker will have the computational power equivalent to the commodity CPUs, we used Google Colaboratory CPU (AMD EPYC 7B12, 2 CPUs @ 2.3 GHz, 13 GB RAM) to execute our attacks and calculate our metrics.

We run every experiment 20 times I don't think this is enough; need to run at least 100 times to minimize the changes in metrics due to random seeds and environmental factors. From the results obtained from this experiment, we reported the average and standard deviation of AR and AETs for each experiment.

add a table describing the key design parameters, their meanings, and their default values used; pick a couple of key design parameters to vary; asymptotic complexity analysis table for various attack algorithms

edited by here

Algorithm 1: Evaluation Experiment

Input: N, A, s_0 specify the meaning of each input parameter
Parameter: targeted, defense, done, PerturbationMethod specify what each parameter means; PerturbationMethod is a function

for each episode **do**
  for each step during an episode **do**
    if targeted is true **then**
      $a_t^{adv*} = \text{RandomStrategy}(A)$
      $s_t^{adv} = \text{PerturbationMethod}(s_t, a_t^{adv*})$
    else
      $s_t^{adv} = \text{PerturbationMethod}(s_t)$
    end if
    if defense is true **then**
      $a_t^{adv} = \text{DQN}_{\text{defense}}(s_t^{adv})$
    else
      $a_t^{adv} = \text{DQN}(s_t^{adv})$
    end if
    $r_t^{adv}, s_{t+1}, \text{done} = \text{Perform}(a_t^{adv})$
    if done is true **then**
      break
    end if
  end for
end for

Experiments & Results

explain why the existing five attacks are selected compared to others. we didn't include Sun's et al., etc.

figures: (1) attack success rates, ASR (non-targeted; targeted; non-targeted under defense 1; targeted under defense 1; non-targeted under defense 2; targeted under defense 2) – (a) table result when #step is fixed at 8; (b) table result when an optimal #step is used at each algorithm; (2) attack execution time per perturbation (non-targeted; targeted) – may vary the values of a key design parameter (e.g., #steps) – two graphs (a) under non-targeted where each curve represents each attack method; (b) same but under targeted attacks; (3) a table summarizing attack/defense reward under non-targeted/targeted, non-targeted/targeted under defense 1, and non-targeted/targeted under defense 2 – total 6 results in a table under various attack methods

We have conducted extensive experiments to test the efficacy and robustness of ER2FGSM. For the purpose of evaluation, we crafted four kinds of attacks for each perturbation attack as follows:

- Non-Targeted Non-Defended Attack
- Targeted Non-Defended Attack

Perturbation Method	No Defense		Defense with SA-DQN		Defense with RANDIAL-DQN	
	Non-targeted	Targeted	Non-targeted	Targeted	Non-targeted	Targeted
CW	-21.00 ± 0.00		$+20.85 \pm 0.36$			
PGD	-21.00 ± 0.00		-20.50 ± 0.50			
DI-FGSM	-21.00 ± 0.00		-19.95 ± 1.16			
MI-FGSM	-21.00 ± 0.00		-20.70 ± 0.46			
FGSM	-21.00 ± 0.00		$+20.75 \pm 0.43$			
ER2FGSM	-21.00 ± 0.00		-20.95 ± 0.22			

Table 2: Comparison of Perturbation Methods in Average Reward (AR)

Perturbation Method	No Defense		Defense with SA-DQN		Defense with RANDIAL-DQN	
	Non-targeted	Targeted	Non-targeted	Targeted	Non-targeted	Targeted
CW	100%	97%	3%			
PGD	100%	100%	100%			
DIFGSM	100%	99%	73%			
MIFGSM	100%	91%	65%			
FGSM	85%	57%	3%			
ER2FGSM	100%	98%	100%			

Table 3: Comparison of Perturbation Methods in Attack Success Rate (ASR)

Perturbation method	AET under non-targeted attacks	AET under targeted attacks
CW	716 ms $\pm$ 32 ms	815 ms $\pm$ 26 ms
PGD	191 ms $\pm$ 7 ms	261 ms $\pm$ 21 ms
DIFGSM	114 ms $\pm$ 5 ms	156 ms $\pm$ 7 ms
MIFGSM	27 ms $\pm$ 2 ms	36 ms $\pm$ 4 ms
FGSM	6 ms $\pm$ 2 ms	6.3 ms $\pm$ 0.9 ms
ER2FGSM	**43** ms $\pm$ 3 ms	**47.1** ms $\pm$ **3.8** ms

Table 4: Comparison Between Different Perturbation Methods Against Attack Execution Time Per Perturbation (AET) Under Targeted or Non-Targeted Attacks.

- Non-Targeted Defended Attack
- Targeted Defended Attack

These four attacks include targeted and non-targeted attacks with or without defense. We define the non-targeted attack as the attack where we just want to generate a perturbation that could reduce the reward instead of particular action. In this non-targeted attack, we select the action that minimizes the reward as much as possible. We define the targeted attack as the attack where we want a special perturbation that could mislead a DRL agent to a desired action. To test our perturbation methods in DRL setting, we gave random actions to perturbation routines in order to test the ability of perturbation method to generate a targeted action. We have designed the Algorithm 1 to show the evaluation experiment.

Comparison w.r.t Attack Success Rate

Needs to be done after complete results, Table 3 and histogram

Comparison w.r.t Average Reward

Needs to be done after complete results, Table 2 and histogram

Comparison w.r.t Attack Execution Time

Needs to be done after complete results, Table 4 and histogram

Comparison when Number of Steps are Constant

To do the fair comparison, we are fixing the number of steps to 8 as ER2FGSM gives optimal performance on 8 number of steps. Needs to be done after results, three histograms maybe or the table?

Parametrization of ER2FGSM

Needs to be done after results, plots/graphs

Number of Steps

Epsilon

Alpha

Conclusion & Future Work

This is the first of its kind research in DRL environment where we highlighted specifically on devising the need of robust and efficient state perturbation attacks by evaluating ER2FGSM with state of the art attacks in terms of attack success rate, attack execution time and robustness.

We obtained the following **key findings** from our study:

- ...
- ...
- ...

For the future work, we plan: (1) using various DRL algorithms and applications to further prove the effectiveness and efficiency of our proposed perturbation attacks; (2) adding more novel perturbation methods by combining the concepts of randomization, momentum, and projected gradient descent; and (3) building novel efficient and defender-aware strategies to attack a defended DRL agent.

References

Alzantot, M.; Balaji, B.; and Srivastava, M. 2018. Did you hear that? adversarial examples against automatic speech recognition. *arXiv preprint arXiv:1801.00554*.

Amarjyoti, S. 2017. Deep reinforcement learning for robotic manipulation-the state of the art. *arXiv preprint arXiv:1701.08878*.

Basori, A. H.; and Malebary, S. J. 2020. Deep reinforcement learning for adaptive cyber defense and attacker's pattern identification. In *Advances in Cyber Security Analytics and Decision Systems*, 15–25. Springer.

Behzadan, V.; and Munir, A. 2017. Vulnerability of Deep Reinforcement Learning to Policy Induction Attacks. *CoRR*, abs/1701.04143.

Carlini, N.; and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, 39–57. IEEE.

Dong, Y.; Liao, F.; Pang, T.; Su, H.; Zhu, J.; Hu, X.; and Li, J. 2018. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 9185–9193.

Ferdowsi, A.; Challita, U.; Saad, W.; and Mandayam, N. B. 2018. Robust deep reinforcement learning for security and safety in autonomous vehicle systems. In *The IEEE 21st International Conference on Intelligent Transportation Systems (ITSC)*, 307–312.

Fischer, M.; Mirman, M.; Stalder, S.; and Vechev, M. 2019. Online robustness training for deep reinforcement learning. *arXiv preprint arXiv:1911.00887*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014a. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014b. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Huang, S. H.; Papernot, N.; Goodfellow, I. J.; Duan, Y.; and Abbeel, P. 2017. Adversarial Attacks on Neural Network Policies. *CoRR*, abs/1702.02284.

Kiran, B. R.; Sobh, I.; Talpaert, V.; Mannion, P.; Al Sallab, A. A.; Yogamani, S.; and Pérez, P. 2021. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*.

Kos, J.; and Song, D. 2017. Delving into adversarial attacks on deep policies. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net.

Kurakin, A.; Goodfellow, I.; and Bengio, S. 2016. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*.

Kurakin, A.; Goodfellow, I.; Bengio, S.; et al. 2016. Adversarial examples in the physical world.

Lin, Y.-C.; Hong, Z.-W.; Liao, Y.-H.; Shih, M.-L.; Liu, M.-Y.; and Sun, M. 2017. Tactics of Adversarial Attack on Deep Reinforcement Learning Agents. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 3756–3762.

Luong, N. C.; Hoang, D. T.; Gong, S.; Niyato, D.; Wang, P.; Liang, Y.-C.; and Kim, D. I. 2019. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Communications Surveys & Tutorials*, 21(4): 3133–3174.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Oikarinen, T.; Weng, T.-W.; and Daniel, L. 2020. Robust deep reinforcement learning through adversarial loss. *arXiv preprint arXiv:2008.01976*.

Pattanaik, A.; Tang, Z.; Liu, S.; Bommannan, G.; and Chowdhary, G. 2017. Robust deep reinforcement learning with adversarial attacks. *arXiv preprint arXiv:1712.03632*.

Sun, J.; Zhang, T.; Xie, X.; Ma, L.; Zheng, Y.; Chen, K.; and Liu, Y. 2020. Stealthy and Efficient Adversarial Attacks against Deep Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04): 5883–5891.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tramèr, F.; Kurakin, A.; Papernot, N.; Goodfellow, I.; Boneh, D.; and McDaniel, P. 2017. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*.

Xie, C.; Zhang, Z.; Zhou, Y.; Bai, S.; Wang, J.; Ren, Z.; and Yuille, A. L. 2019. Improving transferability of adversarial examples with input diversity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2730–2739.

Yoon, S.; Cho, J.-H.; Dixit, G.; and Chen, R. 2021a. Resource-Aware Intrusion Response Based on Deep Reinforcement Learning for Software-Defined Internet-of-Battle-Things. *Game Theory and Machine Learning for Cyber Security*.

Yoon, S.; Cho, J.-H.; Kim, D. S.; Moore, T. J.; Free-Nelson, F.; and Lim, H. 2021b. DESOLATER: Deep Reinforcement Learning-Based Resource Allocation and Moving Target Defense Deployment Framework. *IEEE Access*, 9: 70700–70714.

Yuan, X.; He, P.; Zhu, Q.; and Li, X. 2019. Adversarial Examples: Attacks and Defenses for Deep Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9): 2805–2824.

Zhang, H.; Chen, H.; Xiao, C.; Li, B.; Liu, M.; Boning, D.; and Hsieh, C.-J. 2020. Robust deep reinforcement learning against adversarial perturbations on state observations. *arXiv preprint arXiv:2003.08938*.